

# **TOSHIBA**

Leading Innovation >>>



## **社会インフラシステムへのLinuxの適用**

— Applying Linux to Social Infrastructure Systems —

(株) 東芝 宮川雅紀

2016年 3月 11日

# 自己紹介

---

# 目次

---

- システム概要
- Linux適用で発生した問題の事例
  - 事例1 : pthread\_mutex\_lockによるデッドロック
  - 事例2 : e1000ドライバによるkernelパニック
- まとめ
  - 社会インフラシステムへのLinux適用で見えてきた課題

# 目次

---

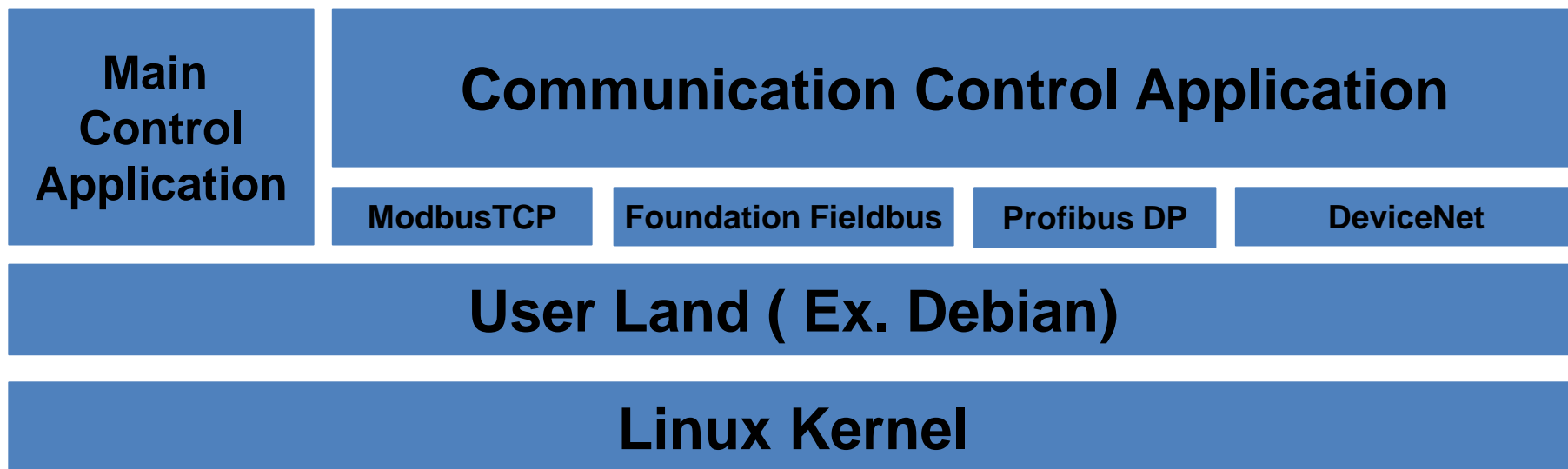
- システム概要
- Linux適用で発生した問題の事例
  - 事例1 : pthread\_mutex\_lockによるデッドロック
  - 事例2 : e1000ドライバによるkernelパニック
- まとめ
  - 社会インフラシステムへのLinux適用で見えてきた課題

# システム概要

- 何故Linuxか？

- TCP/IPベースの産業用通信プロトコルスタックライブラリが入手しやすい
- 豊富なCPUアーキテクチャサポート
- 商用利用可能なディストリビューション
- 産業用通信プロトコルスタックをサードベンダから購入する

- 弊社で開発したシステムの概要は以下



# 目次

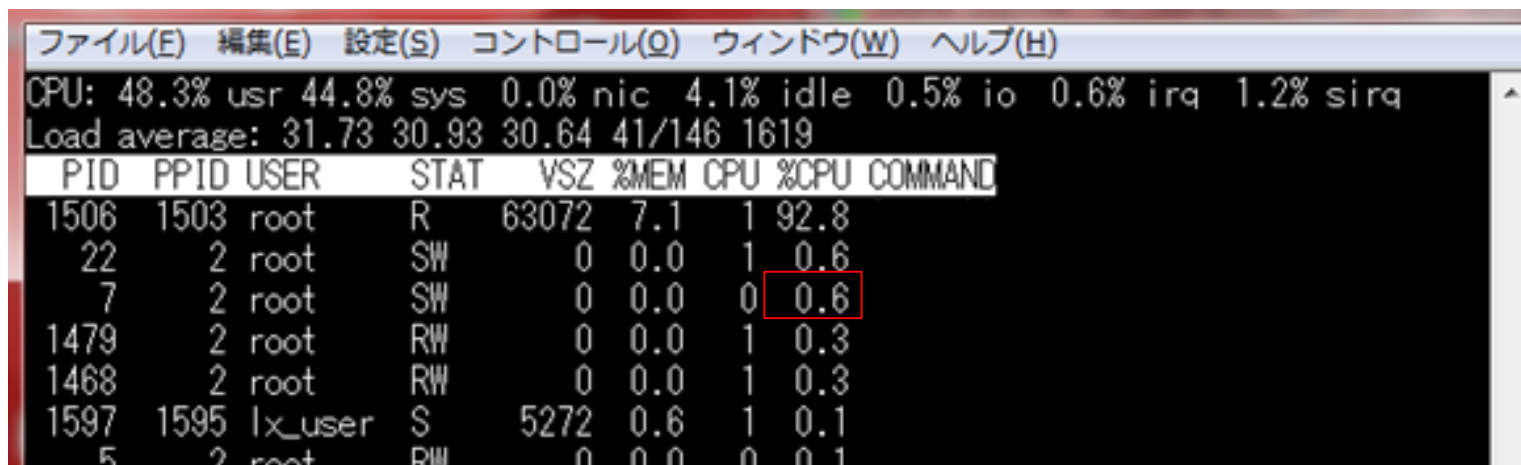
---

- システム概要
- **Linux適用で発生した問題の事例**
  - 事例1 : pthread\_mutex\_lockによるデッドロック
  - 事例2 : e1000ドライバによるkernelパニック
- まとめ
  - 社会インフラシステムへのLinux適用で見えてきた課題

# Linuxを適用して発生した問題の事例

## 事例1

- pthread\_mutex\_lockによるデッドロック
- 問題の概要
  - Linux適用製品の連続通電にて、1週間に2,3回の割合でWDTリセット発生
  - 問題発生時、あるプロセスのCPU使用率が90%を超える
  - 問題を起こすプロセスのスケジューリングポリシーはSCHED\_FIFO (優先度98)



PID	PPID	USER	STAT	VSZ	%MEM	CPU	%CPU	COMMAND
1506	1503	root	R	63072	7.1	1	92.8	
22	2	root	SW	0	0.0	1	0.6	
7	2	root	SW	0	0.0	0	0.6	
1479	2	root	RW	0	0.0	1	0.3	
1468	2	root	RW	0	0.0	1	0.3	
1597	1595	lx_user	S	5272	0.6	1	0.1	
5	2	root	RW	0	0.0	0	0.1	

# Linuxを適用して発生した問題の事例

- `proc/<pid>/task/stat`からスレッドの状態を得る
  - 問題のスレッドより高い優先度を持つスレッド以外スケジュールされない状態

```
/proc/1649/task # cat */stat ; sleep 20; cat */stat;
```

```
1649 (hoge) R 1646 1369 1369 0 -1 4194560 917 0 0 0 4 13 0 0 -2 0 8 < . . . 省略 . . . >
1652 (hoge) R 1646 1369 1369 0 -1 4194624 65 0 0 0 217544 197974 0 0 -3 0 8 < . . . 省略 . . . >
1654 (hoge) S 1646 1369 1369 0 -1 4194368 26 0 0 0 1115 1307 0 0 -2 0 8 0 < . . . 省略 . . . >
1655 (hoge) R 1646 1369 1369 0 -1 4194368 32 0 0 0 462 1026 0 0 -2 0 8 0 < . . . 省略 . . . >
1656 (hoge) R 1646 1369 1369 0 -1 4194368 1 0 0 0 173151 108241 0 0 -2 0 8 0 < . . . 省略 . . . >
1657 (hoge) R 1646 1369 1369 0 -1 4194624 52 0 0 0 83078 35263 0 0 -2 0 8 0 < . . . 省略 . . . >
1658 (hoge) R 1646 1369 1369 0 -1 4194368 43 0 0 0 634972 282472 0 0 -3 0 8 0 < . . . 省略 . . . >
1659 (hoge) R 1646 1369 1369 0 -1 4194368 0 0 0 0 377 1076 0 0 -2 0 8 < . . . 省略 . . . >
```

優先度99のスレッド  
CPU使用時間が更新  
されていない

優先度98のスレッド  
CPU使用時間が更新  
されている

```
1649 (hoge) R 1646 1369 1369 0 -1 4194560 917 0 0 0 4 13 0 0 -2 0 8 0 < . . . 省略 . . . >
1652 (hoge) R 1646 1369 1369 0 -1 4194624 65 0 0 0 218583 198968 0 0 -3 0 8 0 < . . . 省略 . . . >
1654 (hoge) S 1646 1369 1369 0 -1 4194368 26 0 0 0 1115 1307 0 0 -2 0 8 0 < . . . 省略 . . . >
1655 (hoge) R 1646 1369 1369 0 -1 4194368 32 0 0 0 462 1026 0 0 -2 0 8 0 < . . . 省略 . . . >
1656 (hoge) R 1646 1369 1369 0 -1 4194368 1 0 0 0 173151 108241 0 0 -2 0 8 0 < . . . 省略 . . . >
1657 (hoge) R 1646 1369 1369 0 -1 4194624 52 0 0 0 83078 35263 0 0 -2 0 8 0 < . . . 省略 . . . >
1658 (hoge) R 1646 1369 1369 0 -1 4194368 43 0 0 0 636006 283470 0 0 -3 0 8 0 < . . . 省略 . . . >
1659 (hoge) R 1646 1369 1369 0 -1 4194368 0 0 0 0 377 1076 0 0 -2 0 8 0 < . . . 省略 . . . >
```



# Linuxを適用して発生した問題の事例

- **gdbserver + gdbで問題が発生したプロセスにアタッチ**
  - pthread\_mutex\_lockの呼び出しから戻らないことが判明
  - この時ロック変数は優先度99のスレッドによってロックされていた
- **怪しいと思った点**
  - pthread inheritanceを使用していない
  - 優先度99のスレッドと優先度98のスレッドが排他制御を行っている

**優先度逆転現象が起こり得る！**

- **納得できない点**
  - PTHREAD\_MUTEX\_INITIALIZERで初期化していない
    - 優先度逆転現象が起きたとしてもデッドロックはしないはず
  - 常時発生するわけではない

# Linuxを適用して発生した問題の事例

## • 問題の発生をftraceでトレースする

- 短時間でsys\_futexを繰り返し呼び出していることが分かる

```
<function-tracer>
~/dbgfs/tracing # cat trace
# tracer: function
#
#          TASK-PID    CPU#    TIMESTAMP    FUNCTION
#          | |        |         |          |
hoge-1657  [000]    6769.147788: futex_wait_setup <-futex_wait
hoge-1657  [000]    6769.147788: get_futex_key <-futex_wait_setup
hoge-1657  [000]    6769.147789: sys_futex <-syscall_call
hoge-1657  [000]    6769.147789: do_futex <-sys_futex
hoge-1657  [000]    6769.147789: futex_wait <-do_futex
hoge-1657  [000]    6769.147789: futex_wait_setup <-futex_wait
hoge-1657  [000]    6769.147790: get_futex_key <-futex_wait_setup
hoge-1657  [000]    6769.147790: sys_futex <-syscall_call
hoge-1657  [000]    6769.147790: do_futex <-sys_futex
hoge-1657  [000]    6769.147791: futex_wait <-do_futex
hoge-1657  [000]    6769.147791: futex_wait_setup <-futex_wait
hoge-1657  [000]    6769.147791: get_futex_key <-futex_wait_setup
.
```

# Linuxを適用して発生した問題の事例

- **sys\_futexの実装を確認する**

- ロック変数が**4バイト境界に配置されていない時**-EINVALを返す

```
static int
get_futex_key(u32 __user *uaddr, int fshared, union futex_key *key)
{
    unsigned long address = (unsigned long)uaddr;
    struct mm_struct *mm = current->mm;
    struct page *page;
    int err;

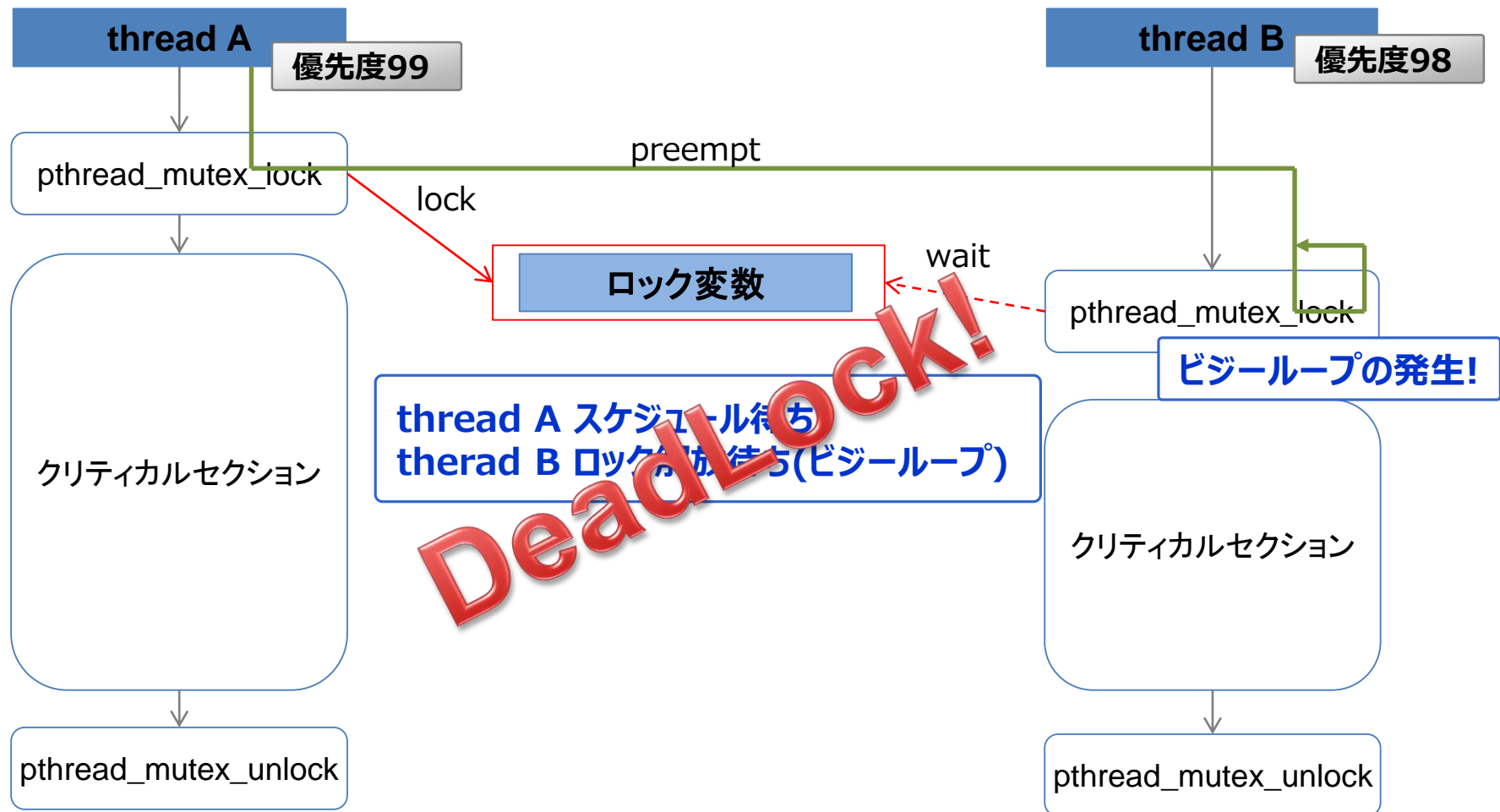
    /*
     * The futex address must be "naturally" aligned.
     */
    key->both.offset = address % PAGE_SIZE;
    if (unlikely((address % sizeof(u32)) != 0))
        return -EINVAL;
    address -= key->both.offset;
```

一方、pthread\_mutex\_lockはロック変数が  
ロックされている状態でsys\_futexから-EINVAL  
が返されたとき、sys\_futexを再度呼び出す実装

**実際にロック変数のアドレスが4バイト境界ではなかった！**

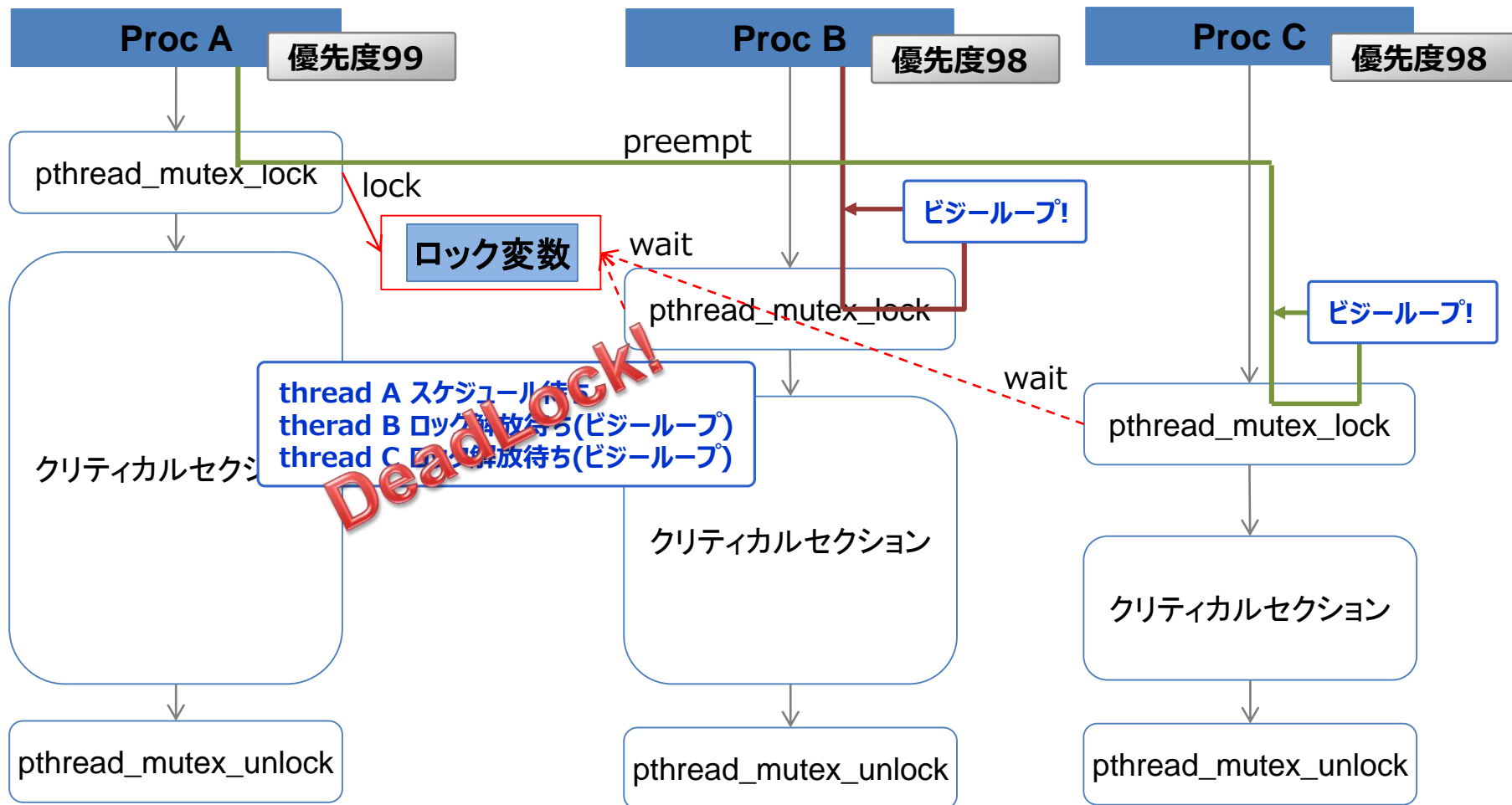
# Linuxを適用して発生した問題の事例

- 問題発生時のpthread\_mutex\_lockの動作まとめ



# Linuxを適用して発生した問題と事例

- マルチコア環境では再現性が低い



# Linuxを適用して発生した問題と事例

- 要因は？

- ロック変数が4バイト境界に配置されていなかったこと
- pthread\_mutex\_lockの実装の問題
- pthread inheritanceを使用していなかったという問題

優先度を継承していれば問題は発生していなかった

- 対策

- ロック変数を4バイト境界に移動
  - pthread\_mutex\_lockの修正は難しい

# Linuxを適用して発生した問題と事例

## • 発生条件まとめ

- ロック変数を4バイト境界に配置しない
- 優先度の異なる複数のリアルタイムスレッドにて排他制御をする
- pthread\_mutex\_lockで排他制御をする
  - pthread inheritanceを使用しない
- 問題を引き起こした環境
  - Kernel : Linux-2.6.33.9-rt31
  - UserLand : Debian Squeeze (eglibc-2.11.3-4)
- 他の比較的新しい環境でも再現
  - Kernel : Linux-3.13.3
  - UserLand : Ubuntu14.04 (glibc 2.19)

# Linuxを適用して発生した問題の事例

## 事例2

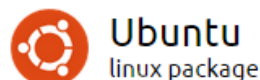
- e1000ドライバによるOops + カーネルパニック
- 問題の概要
  - Linux適用製品の連続通電にて、カーネルパニック発生
  - カーネルが出したバックトレースからe1000ドライバで発生していることを特定
  - 製品出荷後約3年後の出来事



# Linuxを適用して発生した問題の事例

## • 問題に対するパッチはすぐに見つかった

<https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1009545> より引用



Overview Code **Bugs** Blueprints Translations Answers

**BUG: unable to handle kernel NULL pointer dereference at 00000000000000d0; RIP: 0010: [<fffffffffa001a1ea>] [<fffffffffa001a1ea>] e1000\_clean\_tx\_irq+0xfa/0x3e0 [e1000]**

Bug #1009545 reported by rkagan on 2012-06-06

This bug affects 2 people

12

Affects	Status	Importance	Assigned to	Milestone
linux (Ubuntu)	Fix Released	Medium	Unassigned	
Oneiric	Fix Released	Undecided	Brad Figg	

Also affects project Also affects distribution/package Nominate for series

### Bug Description

```
[ 400.216494] BUG: unable to handle kernel NULL pointer dereference at 00000000000000d0
[ 400.220072] IP: [<fffffffffa001a1ea>] e1000_clean_tx_irq+0xfa/0x3e0 [e1000]
[ 400.220072] PGD 36c99067 PUD 36f67067 PMD 0
[ 400.220072] Oops: 0000 [#1] SMP
[ 400.220072] CPU 0
[ 400.220072] Modules linked in: nls_utf8 iso9660 vesafb snd_intel8x0 snd_ac97_codec
ac97_bus snd_pcm snd_timer snd soundcore psmouse snd_page_alloc serio_raw uvccvideo
videodev v4l2_compat_ioctl32 virtio_balloon shpchp lp parport floppy ahci libahci e1000
virtio_pci virtio_ring virtio
[ 400.220072]
[ 400.220072] Pid: 0, comm: swapper Not tainted 3.0.0-20-generic #34-Ubuntu Parallels
Software International Inc. Parallels Virtual Platform/Parallels Virtual Platform
[ 400.220072] RIP: 0010: [<fffffffffa001a1ea>] [<fffffffffa001a1ea>] e1000_clean_tx_irq+
0xfa/0x3e0 [e1000]
[ 400.220072] RSP: 0018:ffff88003fc03d60 EFLAGS: 00010246
[ 400.220072] RAX: 0000000000000000 RBX: ffff88003c13bd90 RCX: 00000000000000d9
[ 400.220072] RDY: 00000000000000d9 RSI: ffff8800004e11e8 RDI: ffff88003ba630f0
[ 400.220072] RBP: ffff88003fc03e10 R08: ffff8800004df000 R09: ffff88003dc000d0
[ 400.220072] R10: 0000000000000001 R11: 0000000000000293 R12: ffff880036e0d600
[ 400.220072] R13: 0000000000000003 R14: ffff8800004df001 R15: 00000000000000d9
[ 400.220072] FS: 0000000000000000 (0000) GS: ffff88003fc00000 (0000) knfs: 0000000000000000
[ 400.220072] CS: 0010 DS: 0000 ES: 0000 CR0: 000000008005003b
[ 400.220072] CR2: 00000000000000d0 CR3: 000000003647b000 CR4: 00000000000000f0
[ 400.220072] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[ 400.220072] ...
```

# Linuxを適用して発生した問題の事例

- 製品出荷から時間が経ち過ぎている

このパッチは製品で使用しているKernelに素直にあたってくれるだろうか・・・

- パッチの内容を確認すると

思ったより小規模、これなら大丈夫！

```
diff --git a/drivers/net/e1000e/e1000.h b/drivers/net/e1000e/e1000.h
index d236efa..a689798 100644 (file)
--- a/drivers/net/e1000e/e1000.h
+++ b/drivers/net/e1000e/e1000.h
@@ -194,6 +194,8 @@ struct e1000_buffer {
    unsigned long time_stamp;
    u16 length;
    u16 next_to_watch;
+   unsigned int segs;
+   unsigned int bytecount;
    u16 mapped_as_page;
};
/* Rx */
```

```
diff --git a/drivers/net/e1000e/netdev.c b/drivers/net/e1000e/netdev.c
index 218e447..ca97be7 100644 (file)
--- a/drivers/net/e1000e/netdev.c
+++ b/drivers/net/e1000e/netdev.c
@@ -642,14 +642,8 @@ static bool e1000_clean_tx_irq(struct e1000_adapter *adapter)
    cleaned = (i == eop);

    if (cleaned) {
-       struct sk_buff *skb = buffer_info->skb;
-       unsigned int segs, bytecount;
-       segs = skb_shinfo(skb)->gso_segs ? 1:
-       /* multiply data chunks by size of headers */
-       bytecount = ((segs - 1) * skb_headlen(skb)) +
-       skb->len;
-       total_tx_packets += segs;
-       total_tx_bytes += bytecount;
+       total_tx_packets += buffer_info->segs;
+       total_tx_bytes += buffer_info->bytecount;
    }

    e1000_put_txbuf(adapter, buffer_info);
@@ -3907,7 +3901,7 @@ static int e1000_tx_map(struct e1000_adapter *adapter,
    struct e1000_buffer *buffer_info;
    unsigned int len = skb_headlen(skb);
    unsigned int offset = 0, size, count = 0, i;
-   unsigned int f;
+   unsigned int f, bytecount, segs;

    i = tx_ring->next_to_use;

@@ -3966,7 +3960,13 @@ static int e1000_tx_map(struct e1000_adapter *adapter,
    }

+   segs = skb_shinfo(skb)->gso_segs ? 1:
+   /* multiply data chunks by size of headers */
+   bytecount = ((segs - 1) * skb_headlen(skb)) + skb->len;
+
    tx_ring->buffer_info[i].skb = skb;
    tx_ring->buffer_info[i].segs = segs;
    tx_ring->buffer_info[i].bytecount = bytecount;
    tx_ring->buffer_info[first].next_to_watch = i;

    return count;
}
```

# Linuxを適用して発生した問題の事例

- 新たな不安

今回は大事を免れたが、今後大きな修正パッチが出た時、無事適用できるか

- Long-term support版を使用する？

- Target kernel selection rules
  - Maintainer will choose one LTS version per year
  - Maintain it for 2 years from its original release

[https://events.linuxfoundation.org/images/stories/slides/elc2013\\_munakata.pdf](https://events.linuxfoundation.org/images/stories/slides/elc2013_munakata.pdf) より引用

- 社会インフラシステムのライフサイクルはもっと長い
  - 20年前に出荷した製品の保守が必要になるケースがある

# まとめ

## • 社会インフラシステムへの適用で見えてきた点

### 事例1より

- 信頼性
  - linuxの信頼性はかなり高い
  - アプリケーションの実装でも思いもよらない落とし穴がある
    - 設計段階で見つけにくい
  - 開発者の経験値は必要

### 事例2より

- 保守性
  - メンテナンス期間はLong-term support版の期間を上回る
  - 社内にOSSの動向をウォッチするメンテナが必須

**TOSHIBA**

**Leading Innovation >>>**